A GUIDE TO NAS-SYS

0.0 Introduction

Imagine the Nascom with no NAS-SYS rom. When switched on, it will start executing whatever instruction it finds in location 0000 and, without a program beginning at this address, the hardware will achieve nothing. Nor can a program be typed in because pressing the keyboard keys will mean nothing to the Nascom since the keyboard itself has to be driven by a program.

To make the hardware perform a useful function, an "operating system" is required and NAS-SYS is an example of just such a small operating system or "monitor". Following power up, NAS-SYS is in control of the hardware. After some initial chores, it scans the keyboard and displays keyed characters on the screen. It then attempts to sort out what the user wishes to do. Does he wish to display or modify the contents of selected ram locations, for example ? Using the various commands, which are described in Chapter 2, the user is given control over the hardware. The combination of the Nascom hardware and the NAS-SYS operating system results in a useful computer system.

One of the major activities of the monitor is to drive the peripherals, which are described from a programmers point of view in Chapter 1. Thus, NAS-SYS includes routines for reading the keyboard, writing to the video display, and transferring data via the serial input and output ports. All these routines and many others are required for the operation of the monitor itself. However, they are also available for use in the users own programs. For example, if the user wishes to read the keyboard during his program, he does not have to write out all the code necessary to do this; instead he simply calls the routine within the monitor. Chapter 3 describes the many useful routines within NAS-SYS and Chapter 4 gives examples of how they might be used.

NAS-SYS is one example of a system program; the BASIC interpreter is another, and several others are available including an assembler, dis-assembler, debugger and compilers or interpreters for various high level languages such as PASCAL and FORTH. All these programs themselves make use of the routines within NAS-SYS. For example, if a BASIC program is requesting an input from the keyboard, it is the keyboard routine within NAS-SYS that is actually being used.

The serious Nascom programmer can greatly extend his comprehension and use of his computer by making the effort to understand NAS-SYS; it is hoped that this book will make that task easier and enjoyable.

1.0 INPUT AND OUTPUT DEVICES

NAS-SYS contains routines for driving the hardware associated with the keyboard, the display screen, and the serial input-output port; it also allows an exchange of data with any other port.

1.1 THE KEYBOARD

The keyboard hardware is physically input/output port 0. These ports are driven by the keyboard driver routine, KBD, within NAS-SYS, see Chapter 3 for a description. (Not all the connections to the ports are used - bit 7 of input port 0 is available to the user via TP3 on Nascom2, and bits 2 and 5 of output port 0 are available via PL3/8 and PL3/6 respectively. Bit 4 is used to drive the TAPE led.)

The keyboard keys are arranged logically in a matrix of 8 rows and 7 columns as shown below:

		Column, CCC						
		6	5	4	3	2	1	0	Mapped location
	1	CH	@	SHFT	CTRL	_	NL	BS	KMAP+8,KMAP
	2	]	R	SP	C	4	V	G	KMAP+7
	3	[	P	1	2	0	/	:	KMAP+6
	4	GRAF	O	Q	3	9	.	;	KMAP+5
Row,	5	--	I	A	W	8	,	L	KMAP+4
RRRR	6	↑	U	S	E	7	M	K	KMAP+3
	7	--	Y	Z	D	6	N	J	KMAP+2
	8	↓	T	X	F	5	B	H	KMAP+1

KMAP=0C01H

The keyboard driver, KBD, scans the rows in turn. When a key is pressed, the input is debounced and the state of the keyboard is entered into the keyboard map, workspace locations KMAP to KMAP+9. Thus, if key P is pressed location KMAP+6 is loaded with 20H, all other map locations being 00.
The position of the pressed key is then encoded as a single byte, structured

SRRRRCCC

where S is the state of the SHFT key,
RRRR is the row number from the above matrix,
CCC is the column number from the above matrix.

Thus, unshifted key P which is at row 3, column 5 produces the byte 0,0011,101 or 1DH.

The keyboard table, KTAB, is then searched to find byte 1DH. It is found 50H bytes down the table which is so ordered that the displacement down the table corresponds to the ASCII code for the key; 50H is the ASCII code for character P.

The keyboard table contains only those characters having ASCII codes 00 to 5FH inclusive. It does not include lower case

characters. If a search fails to find the coded byte, the shift bit,S,is masked off and a second search made through the table. The ASCII code so determined is then modified depending on the state of the Shift, Control, and Graphics keys. The Shift key causes 20H to be added to the ASCII code, so that Shift A becomes 61H, the ASCII code for chracter a. The Control key complements bit 6 of the ASCII code, effectively adding or subtracting 40H, so that Control A produces 01H. Similarly, the Graphics key complements bit 7, effectively adding or subtracting 80H, so that Graphic A produces C1H. By use of the Shift, Control, and Graphics keys in any combination together with the character keys, it is possible to generate all the 256 possible codes from the keyboard.

Note that the effect of the Shift and Graphic keys may be reversed using the K command, see Chapter 2.

If a keypress is detected by the driver routine, KBD, the appropriate ASCII code is returned in register A with the Carry flag set. Otherwise the routine returns with the Carry flag cleared.

The address of the start of the keyboard table, KTAB, is held in workspace location $KTAB (0C6FH) and the length of the table is held in $KTABL (0C6DH). These locations are initialised by NAS-SYS so that the table used is that one within NAS-SYS itself. However, these locations may be changed to point to a user defined table in ram so effectively allowing the codes generated by the keyboard to be determined by the user.

1.2 THE VIDEO DISPLAY

The video display is "memory mapped", ie. it appears to the programmer as a block of read/write memory. It occupies memory locations 0800 to 0BFF, the "video ram".
The screen is divided into sixteen lines, each displaying 48 characters. Each of the 768 (=16 x 48) character positions corresponds to a location in video ram. A map of the screen showing the video ram addresses is shown below.

The character actually displayed at any screen position is determined by the contents of the corresponding video ram location and the character generator roms within the Nascom. The standard Nascom character generator rom is such that to display a particular character its ASCII code should be loaded into the corresponding video ram location. For example, to display 'A' at the beginning of the top line, location 0BCA must be loaded with 41H.

Map of the Video Ram Addresses

```
0BCA(3018) 0BCB 0BCC ...                ..0BF8 0BF9(3065)
080A(2058) 080B 080C ...                ..0838 0839(2105)
084A(2122) 084B 084C ...                ..0878 0879(2169)
088A(2186) 088B 088C ...                ..08B8 08B9(2233)
08CA(2250) 08CB 08CC ...                ..08F8 08F9(2297)
090A(2314)                                     0939(2361)
094A(2378)                                     0979(2425)
098A(2442)                                     09B9(2489)
09CA(2506)                                     09F9(2553)
0A0A(2570)                                     0939(2617)
0A4A(2634)                                     0A79(2681)
0A8A(2698)                                     0AB9(2745)
0ACA(2762)                                     0AF9(2809)
0B0A(2826)                                     0B39(2873)
0B4A(2890)                                     0B79(2937)
0B8A(2954)                                     0BB9(3001)
```

Note: The top line (0BCA to 0BF9) is not scrolled.
Those locations not shown are used as 'margins'.

1.3 THE SERIAL INPUT & OUTPUT PORT

Serial input and output is via a universal asynchronous receiver-transmitter or UART. The transmitter section is output port 1, the receiver section is input port 1, and the control of the UART is via input port 2. The receiver and transmitter are independent of each other except for the number of stop bits which has to be the same for both transmitter and receiver. Link Switch LSW1/5 up produces 1 stop bit while LSW1/5 down produces 2 stop bits. A 110 baud device usually requires 2 stop bits while devices operating at other speeds usually require 1 stop bit.

OUTPUT

The serial output port is driven by driver routine SRLX (Chapter 3).

The serial data is transmitted simultaneously to the cassette as a series of audio tones, to a 20mA loop device as a sequence of 20mA and 0mA, and to an RS232 device as a series of +12 V and -12 V. The speed of transmission may be set to 110, 300, or 1200 baud by means of link switches LSW2/1, LSW2/2, and LSW2/3. Additionally the transmission rate may be set to 2400 baud by adding a link as described below.

Output speed	LSW2/1	LSW2/2	LSW2/3
110 baud	x	up	down
300 baud	down	down	x
1200 baud	up	down	x
External	x	up	up

where x signifies 'don't care'.

The External clock frequency should be 16 times the desired baud rate. For 2400 baud, an 'external' clock of 38.4kHz (=16 x 2400) is available at TP20. Thus, connect TP20 to TP4 (external transmit clock) and set both LSW2/2 and LSW2/3 up.

INPUT

The serial input port is driven by driver routine SRLIN (Chapter 3).

Input data may be accepted from the cassette, from a 20mA loop device, or an RS232 device. Link Switch LSW2/7 determines the source of the data:

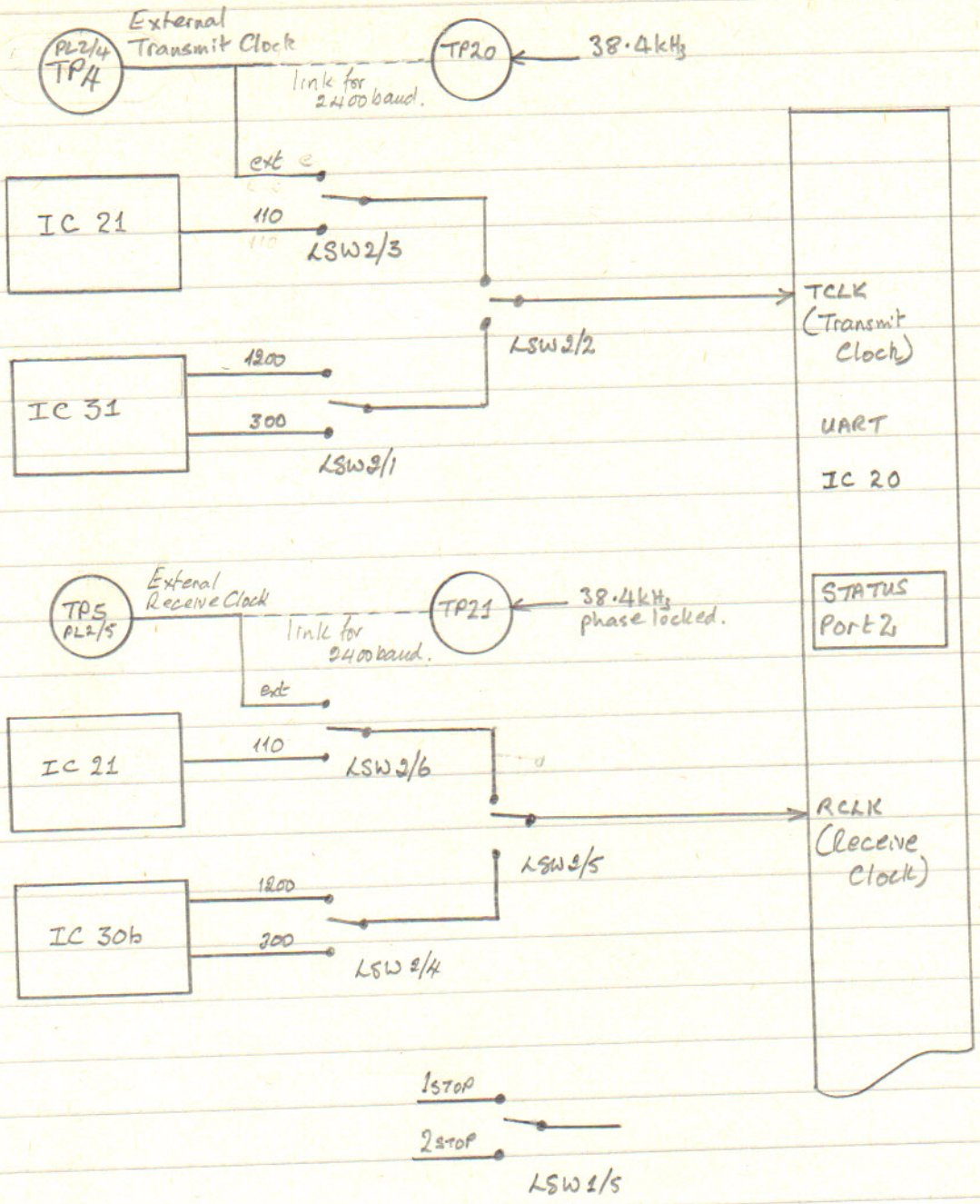
Input from	LSW2/7
Cassette recorder	down
RS232 device or loop device	up

Input speed	LSW2/4	LSW2/5	LSW2/6
110 baud	x	up	down
300 baud	down	down	x
1200 baud	up	down	x
External	x	up	up

where x signifies 'don't care'

The External clock frequency should be 16 times the desired baud rate. For 2400 baud input, an 'external' clock of 38.4kHz (=16 x 2400) which is phase locked to the incoming signal is available at TP21. Thus, connect TP21 to TP5 (external receive clock) and set both LSW2/5 and LSW2/6 up.

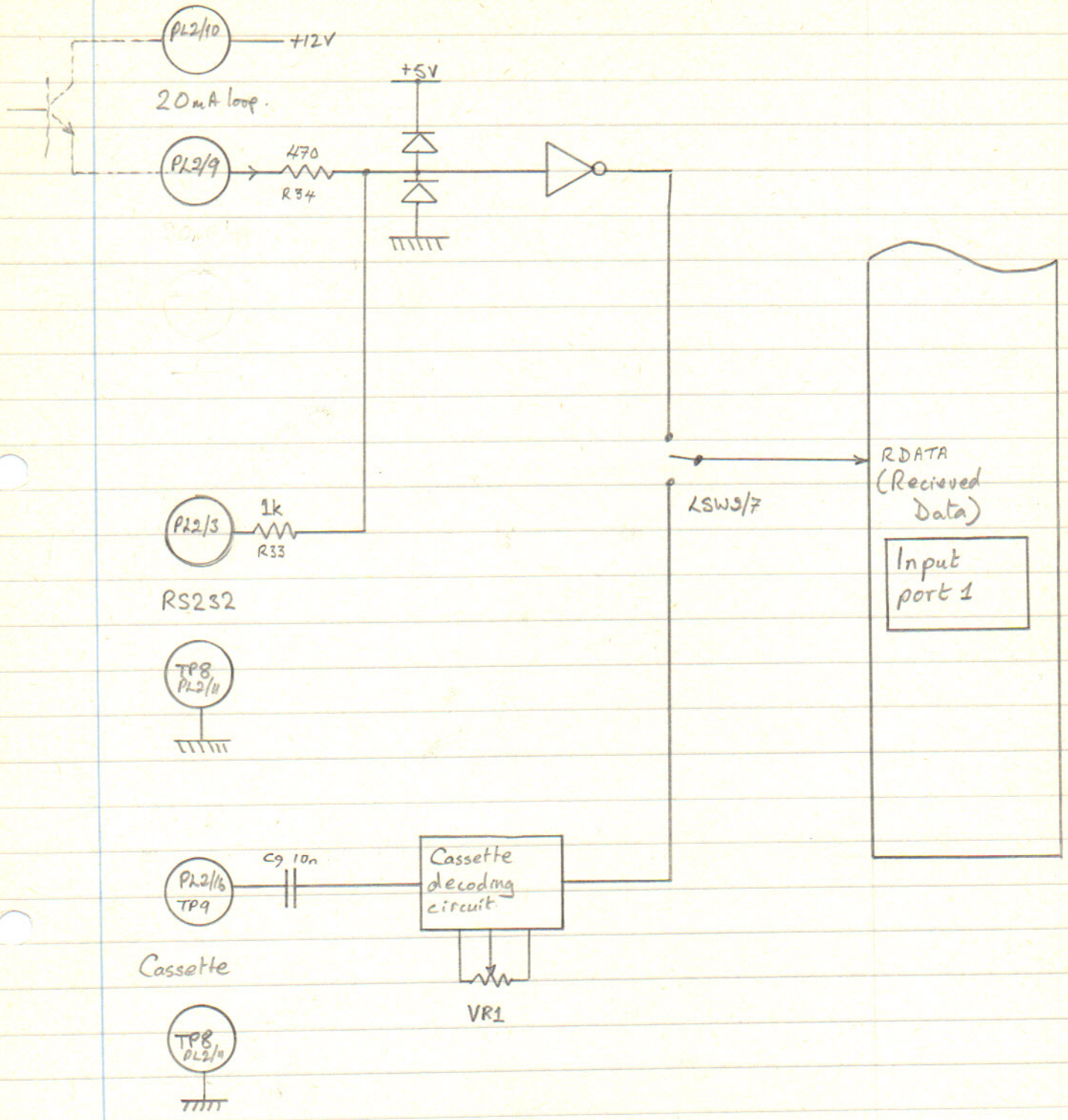
Setting the receive and transmit speeds and the number of stop bits.



The up position of the switches in the diagram corresponds to the up position of the switch itself.

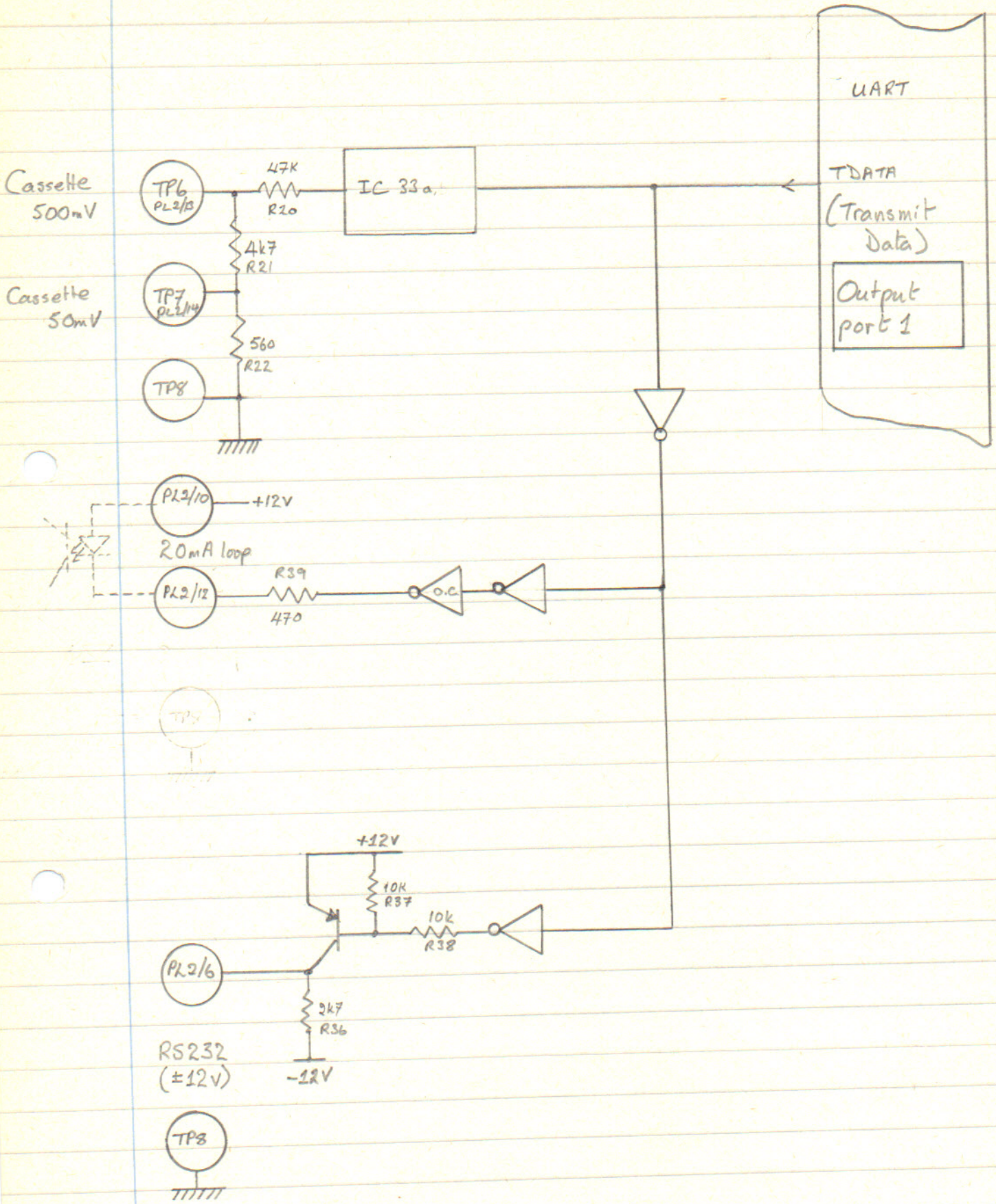
Setting the receive and transmit speeds and the number of stop bits of the serial I/O Port.

Serial input circuits, simplified.



Serial Input Circuits, simplified.

Serial output circuits, simplified



Serial Output Circuits, simplified.

1.4 INPUT AND OUTPUT TABLES

Each input and output device on the Nascom has a device driver, ie. a routine which drives the hardware so as to achieve the required data transfer. Thus, for example, driver SRLIN inputs a byte from the serial input port, input port 1 (the UART receiver), while device driver CRT outputs to the video display. Refer to Chapter 3 for a description of the drivers.

Often data transfers are required to be made to or from more than one device. For example, it is usually convenient for a routine such as BLINK to accept input from either the keyboard or the serial input port. BLINK might have been written so that it always requests input from either of these devices. However, the more flexible method used is for the input and output routines to refer to a table which lists the device drivers to be used by the routine. These tables may be changed by the user at any time so directing input and output to selected devices only.

Following power-up or reset, NAS-SYS initialises workspace location $IN (0C75/6) to 077C (NAS-SYS 3), 0782 (NAS-SYS 1). The following tables show that the input routines will thus read the keyboard and the serial input port. Workspace location $0UT (0C73/4) is similarly initialised to 0779 (NAS-SYS 3), 077F (NAS-SYS 1), thus directing the output routines to write to the video display only.
A subsequent U command changes $IN to 077B (NS3), 0781 (NS1) so adding device driver UIN to the list of input devices, while $0UT is changed to 0778 (NS3), 077E (NS1) so adding device driver U0UT to the list of output devices.
The X command changes $IN to 077F(NS3), 0785 (NS1) so including device driver XKBD, while $0UT is changed to 0777 (NS3), 077D (NS1) so including X0UT, U0UT, and CRT.

OUTPUT TABLES
Address of start of table held in $OUT (0C73/4)

NS1 loc.	NS1 name	NS3 loc	NS3 name	Driver number	Driver			
077A	OUTT2	0774	OUTT2	65	CRT			G
077B	OUTT3	0775	..	6F	SRLX			
077C	..	0776	..	00	End of table.			
077D	OUTTX	0777	OUTTX	6E	XOUT			Xn
077E	OUTTU	0778	OUTTU	75	UOUT		U	
077F	OUTT1	0779	OUTT1	65	CRT	N		
0780	..	077A	..	00	End of table.			

INPUT TABLES
Address of start of table held in $IN (0C75/6)

NS1 loc.	NS1 name	NS3 loc.	NS3 name	Driver number	Driver			
0781	INTU	077B	INTU	76	UIN			U
0782	INT1	077C	INT1	61/7D	KBD/RKBD	N		
0783	INT3	077D	..	70	SRLIN			
0784	..	077E	..	00	End of table.			
0785	INTX	077F	INTX	74	XKBD		X	
0786	INT4	0780		61/7D	KBD/RKBD			
0787	..			00	End of table.			

$UOUT (0C77/8/9) is C3 - - ,jump to user written output driver.
$UIN (0C7A/B/C) is C3 - - ,jump to user written input driver.

1.5 Use of NMI by NAS-SYS

The very useful facility of being able to step through a user program one instruction at a time is achieved by driving the single-step circuit from NAS-SYS. Essentially, the single-step command arms the single-step circuit of the NASCOM so that, after execution of one user program instruction, an NMI (non-maskable interrupt) is generated.
The NMI service routine displays the contents of the cpu registers as they were at the end of the instruction and then waits for a further keyboard command.

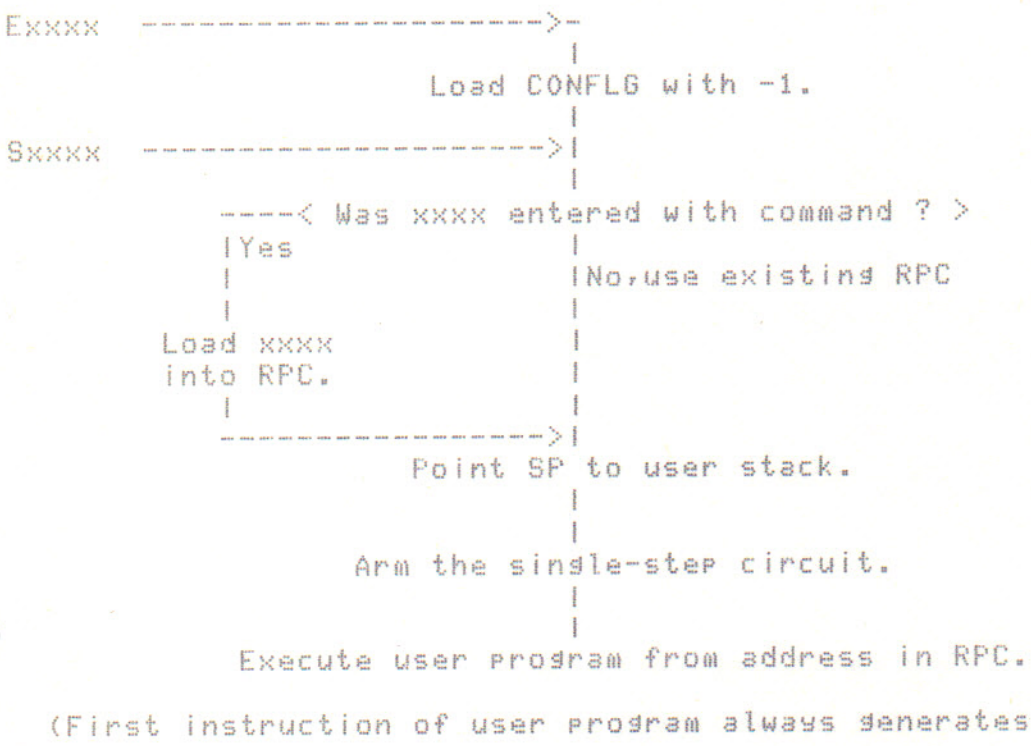
In response to an NMI, the Z80 pushes the program counter onto the stack and then executes the code in location 0066H, which is a JP $NMI instruction. At workspace location $NMI (0C7DH) is the instruction JP TRAP where TRAP is the start of the NMI service routine.

An NMI may also be generated by a push-button which, when pressed, brings connector PL3/4 to zero volts. This button will

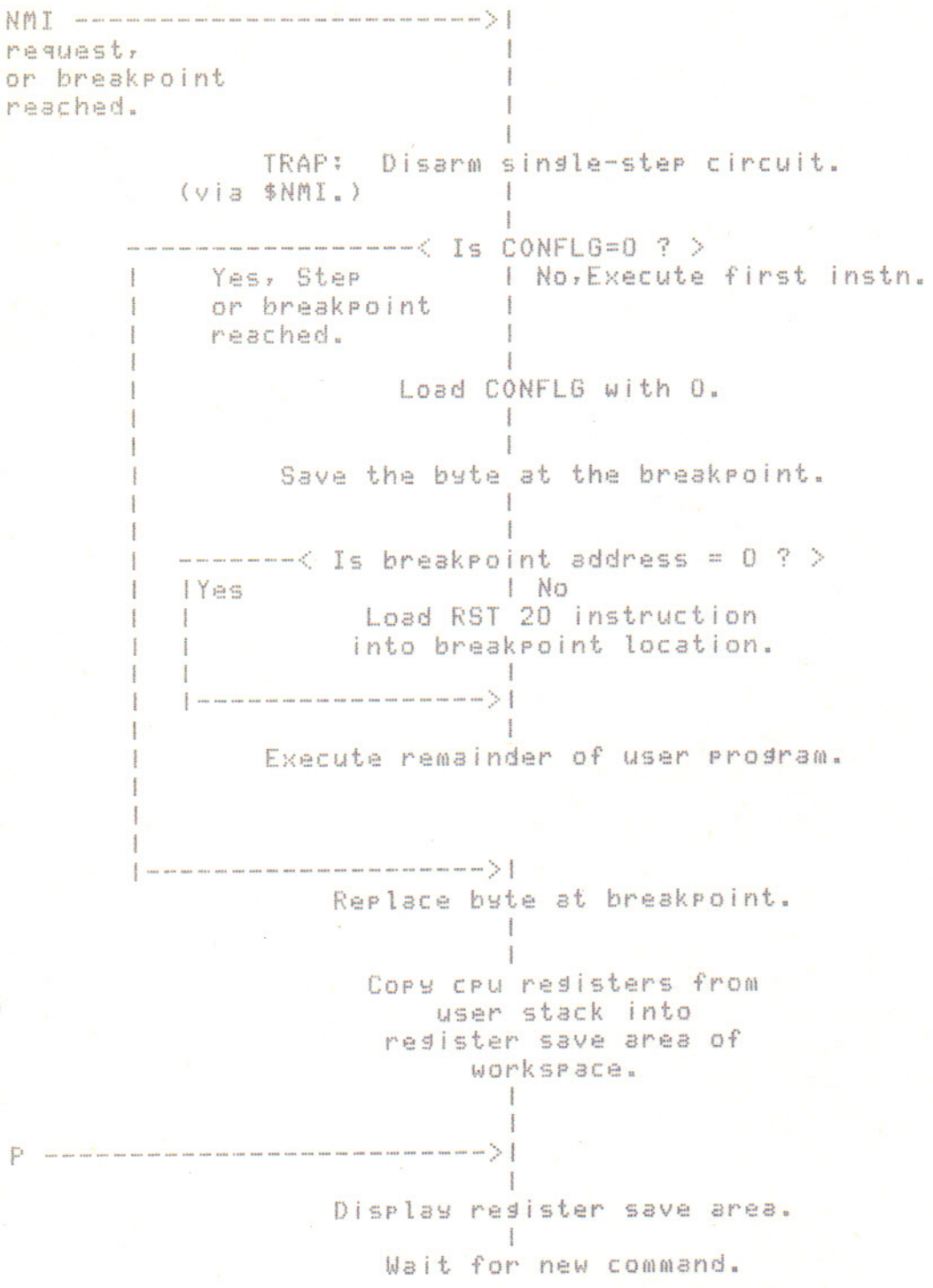
then function as a 'break' key, stopping program execution and displaying the register contents.

A third method of generating an NMI is by driving line 21 of NASBUS from an external device. Such a device will usually require a different response to that of the break key described above. This may be achieved by loading workspace locations $NMI+1/2 with the address of the start of the service routine, which must terminate with a RETN instruction.

The following diagram shows the relationship of the NMI service routine to NAS-SYS 1 commands E and S, which are described in Chapter 3. The NAS-SYS 1 coding differs slightly but performs the same function.



(First instruction of user program always generates an NMI.)


NMI request, or breakpoint reached.
TRAP: Disarm single-step circuit.
(via $NMI.)
Is CONFLG=0 ?
Yes, Step or breakpoint reached.
No, Execute first instn.
Load CONFLG with 0.
Save the byte at the breakpoint.
Is breakpoint address = 0 ?
Yes
No
Load RST 20 instruction into breakpoint location.
Execute remainder of user program.
Replace byte at breakpoint.
Copy cpu registers from user stack into register save area of workspace.
P
Display register save area.
Wait for new command.

1.6 Memory Maps